Attorney Docket No.
SUNMP349/P6983/MG

PATENT APPLICATION

SYSTEM, METHOD AND APPARATUS FOR COMBINING CRYPTOGRAPHIC HASH ALGORITHMS

INVENTORS: Leonard D. Rarick
2817 Amulet Street
San Diego, California 92123
Citizen of United States

Farnad Sajjadian
524 S. Cascade Terrace
Sunnyvale, California 94087
Citizen of United States

Sanjay Patel
33997 Capulet Circle
Fremont, California 94555
Citizen of United States

ASSIGNEE: Sun Microsystems, Inc
4150 Network Circle
Santa Clara, CA 95054

MARTINE & PENILLA, LLP
710 Lakeway Dr., Suite 170
Sunnyvale, California 94085
Telephone (408) 749-6900

SYSTEM, METHOD AND APPARATUS FOR COMBINING CRYPTOGRAPHIC HASH ALGORITHMS

By Inventors:

Leonard D. Rarick, Farnad Sajjadian and Sanjay Patel

BACKGROUND OF THE INVENTION

1. Field of the Invention

[1] The present invention relates generally to hardware cryptographic execution unit, and more particularly, to methods and systems for combining multiple cryptographic algorithms in a single hardware cryptographic execution unit.

2. Description of the Related Art

[2] Data encryption has become commonplace. By way of example, the commonplace e-commerce transactions that occur between customers and merchants on the Internet have driven the demand for efficient, secure interchange of sensitive, personal, financial data between the parties. There are four basic categories of cryptographic algorithm functionality: public key encryption algorithms, bulk encryption algorithms, random number generation algorithms and hashing algorithms. In order to ensure data integrity, several standard cryptographic hash algorithms have been developed and include: MD5 (i.e., Message Digest 5), SHA1 (i.e., secure hash algorithm) and other cryptographic hash algorithms. The MD5 and SHA1 algorithms are described in detail in the "Handbook of Applied Cryptography" by authors Alfred J. Mendezes, Paul C. van Oorschot and Scott A Vanstone, which is incorporated by reference herein for all purposes.

[3] Briefly described, MD5 is a one-way hash function algorithm that is used to create digital signatures. MD5 was designed for use with 32 bit machines and is more

secure than earlier cryptographic algorithms (e.g., MD4 algorithm). A one-way hash function means that MD5 takes a message and converts it into a fixed-length string of digits (i.e., a message digest). The one-way hash function allows a calculated message digest to be compared against the message digest that is decrypted with a public key to verify that the message hasn't been tampered with. This comparison is called a "hashcheck."

[4] The SHA1 was developed for use with the Digital Signature Standard (DSS) and is specified within the Secure Hash Standard (SHS) [National Institute of Standards and Technology (NIST). FIPS Publication 180: Secure Hash Standard (SHS), May 1993, and National Institute of Standards and Technology (NIST). Announcement of Weakness in the Secure Hash Standard. May 1994.]. SHA1 is a cryptographic message digest algorithm similar to the MD4 family of hash functions. SHA1 differs in that it adds an additional expansion operation, one or more additional rounds and the whole transformation was designed to accommodate the DSS block size for efficiency. The SHA1 processes a message to produce- a 160-bit message digest that is designed so that it should be computationally expensive to find a text that matches a given hash. By way of example, given a hash for document A, H(A), it is difficult to find a document B which has the same hash, and even more difficult to arrange that document B says what you want it to say.

[5] Figure 1 shows a typical server 102 and client computer 110 that are linked by a network 104, such as the Internet or other network. Figure 2 is a high-level block diagram of a typical server 102. As shown, the server 102 includes a processor 202, ROM 204, and RAM 206, each connected by a peripheral bus system 208. The peripheral bus system 208 may include one or more buses coupled to each other through various bridges, controllers and/or adapters, such as are well known in the art. For example, the peripheral bus system 208 may include a "system bus" that is connected through an adapter to one or more expansion buses, such as a Peripheral Component Interconnect (PCI) bus. Also coupled to the peripheral bus system 208 are a mass storage device 210, a network interface 212, a number (N) of input/output (I/O) devices 216-1 through 216-N and a peripheral cryptographic processor 220.

[6] I/O devices 216-1 through 216-N may include, for example, a keyboard, a pointing device, a display device and/or other conventional I/O devices. Mass storage device 210 may include any suitable device for storing large volumes of data, such as a magnetic disk or tape, magneto-optical (MO) storage device, or any of various types of Digital Versatile Disk (DVD) or Compact Disk (CD) based storage.

[7] The peripheral cryptographic processor 220 (i.e., crypto-processor) is linked to the processor 202 by the peripheral bus system 208. The crypto-processor 220 includes one or more crypto processing units 228A, 228B. Each of the crypto processing units 228A, 228B is for performing a single crypto algorithm (e.g., MD5 or SHA1). The crypto-processor 220 performs encryption and decryption operations that may be necessary for encrypted data transactions such as between the server 102 and the client 110. In some servers the crypto-processor 220 can also be external to the server 102 and linked to the processor 202 by one of the I/O devices 216-1 through 216-N.

[8] Network interface 212 provides data communication between the computer system and other computer systems on the network 104. Hence, network interface 212 may be any device suitable for or enabling the server 102 to communicate data with a remote processing system (e.g., client computer 110) over a data communication link, such as a conventional telephone modem, an Integrated Services Digital Network (ISDN) adapter, a Digital Subscriber Line (DSL) adapter, a cable modem, a satellite transceiver, an Ethernet adapter, or the like.

[9] Typically the processor 202 can operate at clock speeds of up to or more than 1 GHz. Conversely, the peripheral bus system 208 typically operates at a substantially slower speed such as about 166 MHz or similar. Further, the crypto-processor 220 typically operates at a speed similar to the peripheral bus system 208. This is because the crypto-processor 220 cannot process data any faster than the data can be transported across the peripheral bus system 208. Further, the crypto-processor 220 is typically a customized, specialized processor (i.e. an application specific integrated circuit (ASIC)) that may not be made by the latest, highest performance manufacturing technologies and therefore the maximum processing speed (i.e., the crypto-processor

clock speed) of the crypto-processor 220 is substantially less than the maximum processing speed of the processor 202.

[10] Figure 3 is a flowchart diagram of the method operations 300 of a typical encrypted data transaction within the server 102. The encrypted data transaction can be any data transaction that required encryption, decryption or both encryption and decryption such as an e-commerce transaction between the server 102 and the client computer 110. In operation 305, data is received in the server 102 such as from the client computer 110 or because of a request by the client computer 110.

[11] In operation 310, the received data is analyzed to determine if the received data is encrypted. For example, the data may be encrypted because the data includes a user's personal and/or financial data or other data that is transported during an encrypted session.

[12] If the received data is found to not be encrypted data, in operation 310, then the received data is processed as described in operation 330 below. Alternatively, if, in operation 310, the received data is determined to be encrypted data, then, in operation 315, the encrypted data is sent to the peripheral crypto processor 220 via the peripheral bus system 208.

[13] In operation 320, the crypto processor 220 decrypts the encrypted data. In operation 325, the crypto processor 220 outputs the decrypted data to the processor 202 via the peripheral bus system 208. In operation 330, the processor 202 processes the data to produce result data

[14] In operation 335, the result data is analyzed to determine if the result data should be encrypted. If the result data does not require encryption, then the processor outputs the result data to the client 110, in operation 340, and the method operations end. Alternatively, if, in operation 335, the result data required encryption, then in operation 345, the processor outputs the result data to the crypto-processor via the peripheral bus system 208.

[15] In operation 350, the crypto processor 220 encrypts the result data. In operation 355, the crypto processor 220 outputs the encrypted result data to the

processor 202 via the peripheral bus system 208. In operation 360, the processor outputs the encrypted result data to the client 110 and the method operations end.

[16] Transferring the data to be encrypted, decrypted or processed between the crypto processor 220 and the processor 202 is very slow. Further, the slower processing speed of the crypto processor 220 also limits the rate at which the data is encrypted or decrypted. Further, if a large volume of data such as streaming data (e.g., streaming audio, streaming video, etc.) is being encrypted and/or decrypted then the rate the server 102 can serve the streaming data is limited by the rate at which the streaming data can be encrypted and/or decrypted. Further still, the multiple transfers of the streaming data between the crypto processor 220 and the processor 202 can dominate the usage of the peripheral bus system 208 and the I/O systems inside the crypto processor 220 and the processor 202, thereby limiting further the ability of the processor 202 to perform any functions other than transferring data to and from the crypto processor 220.

[17] One shortfall is that multiple crypto processors 220 may be required to enable the multiple crypto algorithms. By way of example a first crypto processor 220 may include a corresponding crypto processing unit 228A, 228B for each crypto algorithm (e.g., MD5, SHA1). In other words, each of the various crypto algorithms use dedicated crypto processing units to efficiently perform for each crypto algorithm. The crypto processor 220 would determine which crypto processing unit 228A, 228B is needed for the current crypto operation and activate the corresponding crypto processing unit 228A, 228B.

[18] Additionally, the multiple crypto processing units 228A, 228B increase the complexity of the crypto processor 220. By way of example consider the number of full adders required for both crypto processing units 228A, 228B. Further, the multiple crypto processing units 228A, 228B consume large quantities of area of the crypto processor die. As is well known in the art, space on an IC die is very valuable. The space on an IC die must be used as efficiently as possible to minimize processing time and to increase processing power through increased device density on the die. These goals come at an increased cost of design and manufacturing of the IC die.

[19] In view of the foregoing, there is a need for an improved combination crypto algorithm unit that efficiently combines the some of the functionality of multiple crypto algorithm units.

SUMMARY OF THE INVENTION

[20] Broadly speaking, the present invention fills these needs by providing a crypto processing unit capable of performing multiple types of cryptographic operations. It should be appreciated that the present invention can be implemented in numerous ways, including as a process, an apparatus, a system, computer readable media, or a device. Several inventive embodiments of the present invention are described below.

[21] One embodiment provides a crypto algorithm unit. The crypto algorithm unit including a first crypto hash execution module and a second crypto hash execution module. The first crypto hash execution module and the second crypto execution module share multiple components and form a combination crypto algorithm unit. The combination crypto algorithm unit can also include a minimum number of components.

[22] The combination crypto algorithm unit can include multiple muxes. The multiple muxes provide a crypto hash algorithm selection control. The crypto hash algorithm selection control allows the selection of a first subset of the components. The selected first subset of the components can execute a first crypto hash algorithm.

[23] The combination crypto algorithm unit is capable of executing at least two different crypto hash algorithms. The first crypto hash execution module is capable of executing at least one of a group of crypto hash algorithms consisting of an MD5 hash algorithm, a SHA-1 hash algorithm, a SHA256 hash algorithm, a SHA384 hash algorithm, and a SHA512 hash algorithm. The second crypto hash execution module is capable of executing at least one of a group of crypto hash algorithms consisting of an MD5 hash algorithm, a SHA-1 hash algorithm, a SHA256 hash algorithm, a SHA384 hash algorithm, and a SHA512 hash algorithm that is different from the crypto hash algorithm that the first crypto hash execution module is capable of executing.

[24] The combination crypto algorithm unit can be on a single integrated circuit die. The combination crypto algorithm unit and a microprocessor can be combined on a single integrated circuit die.

[25] The combination crypto algorithm unit can include one or more full adders, one or more carry look-ahead adders, and one or more compressors. The one or more compressors can include at least one 4 to 2 compressor.

[26] Another embodiment provides an integrated circuit that includes a microprocessor core and a combination crypto algorithm unit. The combination crypto algorithm unit being coupled to the microprocessor core.

[27] The combination crypto algorithm unit is capable of executing at least two different crypto hash algorithms. The combination crypto algorithm unit can include a first crypto hash execution module and a second hash crypto execution module. The first crypto hash execution module can be capable of executing at least one of a group of crypto hash algorithms consisting of an MD5 hash algorithm, a SHA-1 hash algorithm, a SHA256 hash algorithm, a SHA384 hash algorithm, and a SHA512 hash algorithm. The second crypto hash execution module is capable of executing at least one of a group of crypto hash algorithms consisting of an MD5 hash algorithm, a SHA-1 hash algorithm, a SHA256 hash algorithm, a SHA384 hash algorithm, and a SHA512 hash algorithm that is different from the crypto hash algorithm that the first crypto execution module is capable of executing.

[28] Anther embodiment provides a method of executing a crypto instruction that includes receiving a first crypto hash instruction in a combination crypto algorithm unit, determining a corresponding first crypto hash algorithm for the first crypto instruction, selecting a first set of components in the combination crypto algorithm unit and executing the first crypto hash instruction through the selected first set of components. The method can further include receiving a second crypto hash instruction in the combination crypto algorithm unit, determining a corresponding second crypto hash algorithm for the second crypto hash instruction, selecting a second set of components in the combination crypto algorithm unit, and executing the second

crypto hash instruction through the selected second set of components. The selected second set of components and the selected first set of components sharing a third set of components.

[29] The combination crypto algorithm unit described herein allows a more flexible and compact crypto algorithm unit that is capable of executing two or more types of cryptographic algorithms. This allows for a more compact and space efficient crypto algorithm unit that can even be include on the die with a microprocessor so that processing performance can be further increased.

[30] Other aspects and advantages of the invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrating by way of example the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[31] The present invention will be readily understood by the following detailed description in conjunction with the accompanying drawings, and like reference numerals designate like structural elements.

[32] Figure 1 shows a typical server and client computer that are linked by a network 104, such as the Internet or other network.

[33] Figure 2 is a high-level block diagram of a typical server.

[34] Figure 3 is a flowchart diagram of the method operations of a typical encrypted data transaction within the server.

[35] Figure 4 is a flow diagram of an iteration of a MD5 execution, in accordance with one embodiment of the present invention.

[36] Figure 5 is a flow diagram of an iteration of a SHA-1 execution, in accordance with one embodiment of the present invention.

[37] Figure 6 is a flow diagram of an iteration of a SHA256 execution, in accordance with one embodiment of the present invention

[38] Figure 7 shows a flow diagram of an iteration of a combined MD5 and SHA-1 execution, in accordance with one embodiment of the present invention.

[39] Figure 8 is a flow diagram of the MD5 flow through a combined MD5 and SHA-1 algorithm execution module in accordance with one embodiment of the present invention.

[40] Figure 9 is a flow diagram of the SHA-1 flow through a combined MD5 and SHA-1 algorithm execution module in accordance with one embodiment of the present invention.

[41] Figure 10 shows a flow diagram of an iteration of a combined MD5, SHA-1 and SHA256 execution, in accordance with one embodiment of the present invention.

[42] Figures 11A and 11B are continuations of the flow diagram of an iteration of a combined MD5, SHA-1 and SHA256 execution, in accordance with one embodiment of the present invention.

[43] Figure 12 is a microprocessor die that includes a multiple crypto algorithm execution module, in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION OF THE EXEMPLARY EMBODIMENTS

[44] Several exemplary embodiments for a combination crypto hash algorithm unit will now be described. It will be apparent to those skilled in the art that the present invention may be practiced without some or all of the specific details set forth herein.

[45] An important aspect of secure data communication is to provide for data integrity and message authentication. One aspect of this is the use of cryptographic hash algorithms. There are many such hash algorithms available. Some of the more commonly used hash algorithms are the MD5 and the SHA-1. Recently, additional hash algorithm standards have also been established. The additional hash algorithm standards are referred to as SHA256, SHA384 and SHA512. These additional hash algorithm standards will likely become widely used. Hash algorithms MD5, SHA-1 and SHA256 use 32-bit computations where hash algorithms SHA384 and SHA512 use 64-bit computations.

[46] One module can be created to execute two or more of the hash algorithms. By way of example, a single module can execute both MD5 and SHA-1. A second module can execute SHA256, SHA384 and SHA512. A third module can execute MD5, SHA-1 and SHA256. Combining multiple hash algorithms in a single execution module can save significant area on an integrated circuit as compared to having multiple, dedicated execution modules where each hash algorithm has a corresponding dedicated execution module. Performance is not substantially affected in the combined execution modules. The area savings can be achieved by sharing the flip-flops, carry-save adders, carry look-ahead adders and the execution of some functions. To accomplish this some additional multiplexers (muxs) can also be used. Additional control functions may also be implemented.

[47] The MD5 algorithm uses four chaining variables: H1, H2, H3 and H4. The MD5 algorithm also uses four working variables: A, B, C and D. The chaining variables and working variables are 32 bits each. Before processing, the MD5 algorithm must be initialized. The MD5 algorithm hashes sixteen 32-bit words: X[0]

through X[15] at a time. Each hashing of sixteen words requires sixty-four iterations. For each iteration "j", the following is executed:

t = A+ function [j](B, C, D) + X[z[j]] + Y[j]
next A = D
next B = B + rotated [j](t)
next C = B
next D = C

[48] Where z[j] selects one of the input X values and each Y[j] is a specified 32-bit constant. The function performed and the amount t is rotated by depends on the iteration j. Four functions are used:

(C and B) or (D and not B) is used for the first 16 iterations (iterations 0-15)
(B and D) or (C and not D) is used for the second 16 iterations (iterations 16-31)
B xor C xor D is used for the third 16 iterations (iterations 32-47)
C xor (B or not D) is used for the fourth 16 iterations (iterations 48-63)

[49] After the 64 iterations, the chaining variables H1 – H4 are updated as follows:

next H1 = A + H1
next H2 = B + H2
next H3 = C + H3
next H4 = D + H4

[50] It should be understood that the chaining variables H1 – H4 can be updated as the respective working variables A, B, C and D become available, rather than delaying updating until after the 64 iterations are completed.

[51] After the last 512-bit block has been processed, the hash value is the 128-bit concatenation of the chaining variables H1 - H4.

[52] Figure 4 is a flow diagram 400 of an iteration of a MD5 execution, in accordance with one embodiment of the present invention. Working variables B, C, and D are input to the MD5 function block 405. An output of the MD5 function block 405 is input to a first add block 410 with working variable A and values X and Y. An output t from the first add block 410 is input to a rotate block 415. The output of the rotate block 415 is input to a second add block 420 with the working variable B. The output of the second add block is the next B working variable. The B working variable becomes the next C working variable. The C working variable becomes the next D working variable. The D working variable becomes the next A working variable. The X value(s) are retained.

[53] The SHA-1 algorithm uses five chaining variables: H1-H5 and five working variables: A-E. The chaining variables and working variables are 32 bits each. Before processing, the SHA-1 algorithm is initialized. The SHA-1 algorithm hashes sixteen 32-bit words, X[0] through X[15] at a time. Each hashing of sixteen words requires 80 iterations. For each iteration "j", the following are executed:

t = rotated_27(A) + function [j](B, C, D) + E + X[j] + Y[j]
next A = t
next B = A
next C = rotated_2 B
next D = C
next E = D

[54] Where X[j] = rotated_31 (X[j-3] XOR X[j-8] XOR X[j-14] XOR X[j-16]) for j > 15. Each Y[j] is a specified 32-bit constant. The Y[j] constants are different in SHA-1 from MD5. Rotations are right by the amount specified. Three different functions are used:

(C and B) or (D and not B) are used for the first 20 iterations (iterations 0-19).

(B and C) or (C and D) or (D and B) are used for the third 20 iterations (iterations 40-59).

B XOR C XOR D are used for the second and fourth 20 iterations (iterations 20-39 and iterations 60-79)

[55] After 80 iterations, the chaining variables are updated as follows:

next H1 = A + H1

next H2 = B + H2

next H3 = C + H2

next H4 = D + H4

next H5 = E + H5

[56] After the last 512-bit block, the hash value is a 160-bit concatenation of the chaining variables H1-H5.

[57] Figure 5 is a flow diagram 500 of an iteration of a SHA-1 execution, in accordance with one embodiment of the present invention. Working variables B, C and D are input to the SHA-1 function block 505. An output of the SHA-1 function block 505 is input to a add block 510 with working variable E and values X and Y. Working variable A is rotated in first rotate block 515 and the rotated output is input to the add block 510. An output t from the add block 510 becomes the next A. Working variable B is rotated in second rotate block 520 and the rotated output is the next C. Working variable A becomes the next B. Working variable C becomes the next D. Working variable D becomes the next E. A new X value is obtained from the previous sixteen X values in the expand block 255. The expanded output is the new X value.

[58] The SHA256 algorithm uses eight chaining variables: H1-H8, and eight working variables: A-H. The chaining variables H1-H8 and working variables A-H are 32-bits each. Before processing, the SHA256 algorithm is initialized. The SHA256 algorithm hashes sixteen 32-bit words: X[0] through X[15] at a time. Each hashing of sixteen words requires sixty-four iterations. For each iteration "j", the following are executed:

RE = rotated_6 (E) XOR rotated_11 (E) XOR rotated_25 (E)

RA = rotated_2 (A) XOR rotated_13 (A) XOR rotated_22 (A)

CH = (F and E) or (G and not E)

MAJ = (A and B) or (B and C) or (C and A)

t1 = H + RE + CH + X[j] + Y[j]

t2 = RA + MAJ

next A = t1 + t2

next B = A

next C = B

next D = C

next E = D + t1

next F = E

next G = F

next H = G

Where X[j] = R[j – 2] + X[j – 7] + S[j – 15] + X[j – 16] for j > 15, and

R[j – 2] = rotated_17 (X[j – 2]) XOR rotated_19 (X[j – 2]) XOR shifted_10 (X[j – 2]), and

S[j – 15] = rotated_7 (X[j – 15]) XOR rotated_18 (X[j – 15]) XOR shifted_3 (X[j – 15]).

[59] Each Y[j] is a specified 32-bit constant. The Y[j] constants are different than in SHA-1 and MD5 described above. Rotations are right by the amount specified. Shifts are to the right. After 64 iterations, the chaining variables H1-H8 are updated as follows:

next H1 = A + H1

next H2 = B + H2

next H3 = C + H3

next H4 = D + H4

next H5 = E + H5

next H6 = F + H6

next H7 = G + H7

next H8 = H + H8

[60] After the last 512-bit block, the hash value is a 256-bit concatenation of the chaining variables H1-H8.

[61] To increase the execution speed, three carry look a head adders are used. For each iteration the following are computed substantially simultaneously:

t1 + t2
D + t1
new X[j]

[62] An exemplary carry save adder structure can include the following:

P0, P1 = full adder of (X[j], Y[j], H)
P2, P3 = full adder of (MAJ, RA, ZH1)
P4, P5 = 4 to 2 compressor of (P0, P1, RE, CH)
P6, P7 = 4 to 2 compressor of (P4, P5, D, ZH5)
P8, P9 = 4 to 2 compressor of (P2, P3, P4, P5)
Q0, Q1 = 4 to 2 compressor of (R[j – 2], X[j – 7], S[j – 15], X[j – 16])

[63] Where ZH1 = 0 except for the last iteration. ZH1 = H1 for the last iteration. ZH5 = 0 except for the last iteration. ZH5 = H5 for the last iteration. The carry look-ahead addition of P6 and P7 produces the t1 and D values. The carry look-ahead addition of P8 and P9 produces the t1 + t2 values. The carry look-ahead addition of Q0 and Q1 produces the new X[j] value. Zh1 and Zh5 are used to produce two of the updated chaining values during the last iteration (j = 63). Once the value of j is within three of the last value of j, no new X[j] values need be computed. As a result, the carry look-ahead adder can be used to compute three more of the updated chaining values, one at a time. Thus, by the time the last iteration is finished, five of the updated chaining values have already been computed. Then the three remaining chaining

values to be updated may be computed at the same time, one in each of the three carry look-ahead adders.

[64] Figure 6 is a flow diagram 600 of an iteration of a SHA256 execution, in accordance with one embodiment of the present invention. Working variable A becomes the next B. Working variable A is also input to a first rotate and XOR block 605. The rotated and XORd output from the first rotate and XOR block 605 is input to a first add block 615. Working variable B becomes the next C. Working variables A, B and C are input to a majority block 610. The output of the majority block 610 is also input to the first add block 615. T2 is output from the first add block 615 and input to the second add block 620. A second input to the second add block 620 is provided by a third add block 640 that will be described in more detail below. The second add block 620 outputs the next A working variable. Working variable C becomes the next D.

[65] Working variable D is input to a fourth add block 625. A second input to the fourth add block 625 is provided by a third add block 640. The fourth add block 625 outputs the next E working variable. Working variable E becomes the next F. Working variable F becomes the next G. Working variable G becomes the next H. Working variable E is also applied to the input of a second rotate and XOR block 630 and to the input of a CH block 635. The output of the second rotate and XOR block 630 is applied to an input to the third add block 640. Working variables F and G are also applied to the input of the CH block 635. The output of the CH block 635 is applied to an input to the third add block 640. Working variable H and the X and Y values are also applied to respective inputs to the third add block 640.

[66] Several of the X values are also input to a rotate and shift block 645 and to two inputs of a fifth add block 650. The two outputs of the rotate and shift block 645 are also applied to two inputs of the fifth add block 650. The output of the fifth add block 650 is a new X value. After the last 512-bit block, the hash value is a 256-bit concatenation of the chaining variables H1-H6.

[67] The SHA512 algorithm is similar to SHA256. One significant difference is that SHA512 uses 64-bit values (e.g., for the input X values, the chaining variables and the working variables) while SHA256 uses 32-bit values. SHA512 also uses different constants. SHA512 also uses different amounts for the various rotates and shifts. SHA512 also performs eight iterations as compared to sixty-four iterations performed by SHA256.

[68] After the last 1024-bit block, the hash value is a 512-bit concatenation of the chaining variables H1-H8. To increase the execution speed similar considerations for the carry save and carry look-ahead adders as described above for SHA256 can be applied.

[69] A SHA384 algorithm is substantially identical to the SHA512 algorithm except that the initialization constants for H1-H8 are different. Further the hash value is the concatenation of H1-H6. Thus, H7 and H8 are computed but not used in the hash value in SHA384.

[70] MD5 and SHA-1 algorithms can be combined to save significant area of the integrated circuit as described above. Multiple multiplexers (muxes) can be used to allow the combination of MD5 and SHA-1 algorithms in one execution module. The muxes can provide controls to one or more of the various components (e.g., 4 to 2 compressors, adders and flip-flops).

[71] The MD5 and SHA-1 algorithms can be combined into one execution module. A combination of MD5 and SHA-1 algorithms uses five chaining variables: H1-H5, and five working variables: A-E. For each iteration:

for MD5: S1, S0 = 4 to 2 compressor of (A, function[j] (B, C, D), X[z[j]], Y[j])
for SHA-1: S1, S0 = 4 to 2 compressor of (rotated_27(A), function[j] (B, C, D), X[j], Y[j])

[72] Either A or the rotated_27(A) is muxed into one input of the 4 to 2 compressor. For the same value of j, sometimes the function[j]'s are the same and sometimes not for MD5 and SHA-1. All functions are computed. The appropriate function, depending on the value of j and whether MD5 or SHA-1 is being executed, is muxed into the second input of the 4 to 2 compressor. Likewise for the third and fourth inputs to the 4 to 2 compressor, the appropriate value for X and constant Y[j] are muxed. The outputs S0 and S1 of the 4 to 2 compressor are then input into a carry look-ahead adder. The output of the carry look-ahead adder is denoted as "ta." Once ta is computed, then the following can be computed:

for MD5: tb = B + rotated[j](ta)

for SHA-1: tb = E + ta

[73] The muxes are used to select the appropriate inputs to the carry look-ahead adder. Then the working variables are updated:

for MD5	for SHA-1
next A = D	next A = tb
next B = tb	next B = A
next C = B	next C = rotated_2(B)
next D = C	next D = C
next E = D	next E = D

[74] The muxes are used to select the appropriate update values, except for next D and next E. For MD5, 64 iterations are needed, where 80 iterations are used for SHA-1.

[75] Figure 7 shows a flow diagram 800 of an iteration of a combined MD5 and SHA-1 execution, in accordance with one embodiment of the present invention. Working variable A is input to a first rotate block 802 and a first input of a first mux 804. The output of the first rotate block 802 is applied to a second input of the first mux 804. Five functions of the working variables B, C and D that are used by MD5

and SHA-1 are all computed (not shown). These five function values are input to a second mux 806. The output of the first mux 804 and the second mux 806 are applied to a first 4 to 2 compressor 808. Values X and Y are also input to the first 4 to 2 compressor 808. The two outputs S0 and S1, of the first 4 to 2 compressor 808, are applied to a first carry look-ahead adder 810. The output ta of the first carry look-ahead adder 810 is applied to a second rotate block 812 and a third mux 814. The output of the second rotate block 812 is also applied to the third mux 814. The output of the third mux 814 is applied to the second carry look-ahead adder 820.

[76] The working variables B and D are applied to a fourth mux 816. The output of the fourth mux 814 is applied to the second carry look-ahead adder 820. The output tb of the second carry look-ahead adder 820 is applied to a fifth mux 830 and a sixth mux 832. Working variable D is also input to the fifth mux 830. The output of the fifth mux 830 is the next A. Similarly, working variable A is also applied to the sixth mux 832. The output of the sixth mux 832 is the next B.

[77] The working variable B is also applied to a third rotate block 822 and a seventh mux 834. The output of the third rotate block 822 is applied to the seventh mux 834. The output of the seventh mux 834 is the next C. Working variable C is also output as the next D. Working variable D is also output as the next E.

[78] Figure 8 is a flow diagram 900 of the MD5 flow through a combined MD5 and SHA-1 algorithm execution module 800 in accordance with one embodiment of the present invention. Four functions of working variables B, C and D that are used by MD5 are computed (not shown) and are input to the second mux 806. The output of the second mux 806 is applied to a first 4 to 2 compressor 808. Values X and Y and working variable A are also input to the first 4 to 2 compressor 808. The two outputs S0 and S1 of the first 4 to 2 compressor 808 are applied to a first carry look-ahead adder 810. The output ta of the first carry look-ahead adder 810 is applied to a second rotate block 812. The output of the second rotate block 812 is applied to the second carry look-ahead adder 820. Working variable B is also applied to the second carry look-ahead adder 820. The output tb of the second carry look-ahead adder is the next B. Working variable D is output as the next A and working variable B is output as the

next C. Working variable C is output as the next D. The various blocks not used by the MD5 flow can be shut off or otherwise bypassed by the various muxes.

[79] Figure 9 is a flow diagram 1000 of the SHA-1 flow through a combined MD5 and SHA-1 algorithm execution module 800 in accordance with one embodiment of the present invention. Working variable A is input to a first rotate block 802. The output of the first rotate block 802 is applied to a first 4 to 2 compressor 808. Three functions of working variables B, C and D that are used by SHA-1 are computed (not shown) and are input to a second mux 806. The output of the second mux 806 is also applied to the first 4 to 2 compressor 808. Values X and Y are also input to the first 4 to 2 compressor 808. The two outputs S0 and S1 of the first 4 to 2 compressor 808 are applied to the first carry look-ahead adder 810.

[80] The output ta of the first carry look-ahead adder 810 is applied to a second carry look-ahead adder 820. The working variable E is also applied to the second carry look-ahead adder 820. The output tb of the second carry look-ahead adder 820 is the next A. Working variable A is also output as the next B. Working variable B is rotated in the third rotate block 822 and the rotated B is output as the next C. Working variable C is also output as the next D and working variable D is also output as the next E.

[81] It should be noted that while various components (e.g., flip-flops, muxes, etc.) are not shown in Figures 8-10, it should be understood that these various components are included in the figures. By way of example, if an iteration requires more than one clock, then additional flip-flops may be required. If these additional flip-flops can be used in both the MD5 and SHA-1 executions, then the common use can yield additional area/complexity savings to a combined MD5/SHA-1 execution module on a single IC die. Further, computing X and Y values for each iteration is also not shown in the figures. Since the MD5 and the SHA-1 methods for computing X and Y values are different, then there may be no savings in this portion of a combined MD5/SHA-1 execution module. Updating the chaining variables H1-H5 are similarly not shown. The muxing to use the existing carry look-ahead adders and saving the results can be used by both MD5 and SHA-1, thereby saving many muxes as compared to

implementing both of the MD5 and the SHA-1 algorithms separately. While the control of a combined MD5/SHA-1 execution module can be more complex, the area required on the IC die is likely smaller than the area required for separate control circuits for the two separate algorithms. Further, control circuitry is typically smaller than the components required for data flow.

[82] SHA256, SHA384 and SHA512 algorithms can also be combined to yield similar IC die area savings. These algorithms are very similar. Therefore, combining the SHA256, SHA384 and SHA512 algorithms into one module can be accomplished somewhat easily. The SHA256 algorithm uses 32-bit values whereas the SHA384 and SHA512 algorithms use 64-bit values. In some instances, the most significant 32-bits of the 64-bit values used by the SHA512 algorithm are the values used by the SHA256 algorithm. As a result, 64-bit values are used for the working variables and the chaining variables. For SHA256, the least significant 32-bits are kept to zero. Muxing the appropriate initialization values and selecting the correct rotation and shifting, then one module can execute any of the SHA256, SHA384 and SHA512 algorithms.

[83] The MD5, SHA1 and SHA256 algorithms can also be combined as each uses 32-bit words, so all values are only 32-bits. There are no 64-bit values in these three algorithms. The carry save adder structure described above for the SHA256 is used for the MD5, SHA1 and SHA256 algorithm combination module. Controlling the inputs to the various components (e.g., full adders, 4 to 2 compressors, adders, and flip-flops) the values needed for MD5 and SHA-1 can be obtained instead of the SHA256 values. For the MD5 algorithm, the inputs are:

P1, P0 = full adder of (X[z[j]], Y[j], zero)
P3, P2 = full adder of (function, A, zero)
P5, P4 = 4 to 2 compressor of (P0, P1, zero, zero)
P9, P8 = 4 to 2 compressor of (P2, P3, P4, P5)

[84] The carry look-ahead addition of P8 and P9 produce the t value. The rotated t value and B are then added in a carry look-ahead adder to produce the next B value.

The MD5 algorithm may be slower than the other algorithms as the two carry look-ahead additions are carried out sequentially instead of simultaneously. During the last iteration, while the t value is being computed in one carry look-ahead adder, a second carry look-ahead adder can add B to H2. The resulting sum, rather than B, can be added to the rotated t value to determine the updated H2 value. A third carry look-ahead adder may not be needed during the iterations and therefore, the third carry look-ahead adder may be used to update the other three chaining variables H1, H3, and H4, during the last few iterations. As a result, no work needs to be done after the last iteration is finished to obtain the updated chaining variables.

[85] For the SHA-1 algorithm, the inputs are:

P1, P2 = full adder of (X[j], Y[j], zero)
P3, P2 = full adder of (function, rotated (A), ZH1)
P5, P4 = 4 to 2 compressor of (P0, P1, E, zero)
P9, P8 = 4 to 2 compressor of (P2, P3, P4, P5)

[86] Where ZH1 = zero except for the last iteration. ZH1 = H1 for the last iteration. The carry look-ahead addition of P8 and P9 produces the t value. A third carry look-ahead adder is not needed during the iterations, so it maybe used to update the other four chaining variables (H2, H3, H4, and H5) during the last four iterations. As a result, no work may be required after the last iteration is completed to obtain the updated chaining variables.

[87] Figures 10, 11A and 11B show a flow diagram 1100 of an iteration of a combined MD5, SHA-1 and SHA256 execution, in accordance with one embodiment of the present invention. Working variable A is input to a first rotate block 1102 and a first input of a first mux 1104. The output of the first rotate block 1102 is applied to a second input of the first mux 1104. RA is also applied to a third input of the first mux 1104. Five functions of working variables B, C and D that are used by MD5 and SHA-1 are computed (not shown). These five functions of the working variables B, C and D and a majority function are input to a second mux 1106. The output of the first mux

1104 and the second mux 1106 are applied to a first full adder 1120. Chaining variable H1 is applied to a first AND gate 1118 with a last iteration and not MD5 control bit. The output of the first AND gate 1118 is also input to the first full adder 1120.

[88] Working variable H and a SHA256 control bit are applied to a second AND gate 1108. The output of the second AND gate 1108 is applied to a second full adder 1112. Values X and Y are also applied to the second full adder 1112. The two outputs of the second full adder are applied to a first 4 to 2 compressor 1122. RE and working variable E are applied to a third mux 1110. The output of the third mux 1110 is applied to a third AND gate 1114 along with a not MD5 control bit. The output of the third AND gate 1114 is applied to the first 4 to 2 compressor 1122. CH and a SHA256 control bit are applied to a fourth AND gate 1116. The output of the fourth AND gate 1116 is also applied to the first 4 to 2 compressor 1122.

[89] The two outputs of the first full adder 1120 is applied to a second 4 to 2 compressor 1130. The two outputs of the first 4 to 2 compressor 1122 are applied to the inputs of the second 4 to 2 compressor 1130 and the third 4 to 2 compressor 1136. The outputs of the second 4 to 2 compressor 1130 is applied to a fourth mux 1150 and a fifth mux 1152. Working variables C and F are applied to a sixth mux 1128. The output of the sixth mux 1128 is also applied to the fourth mux 1150. Chaining variables H1 and H6 are applied to a seventh mux 1132. The output of the seventh mux 1132 is also applied to the fifth mux 1152. The outputs of the fourth mux 1150 and the fifth mux 1152 are applied to a first carry look-ahead adder 1160. The output of the first carry look-ahead adder 1160 is applied as shown in Figures 12A and 12B below.

[90] Chaining variable H5 is applied to a fifth AND gate 1124 with a last iteration control bit. The output of the fifth AND gate 1124 is applied to the third 4 to 2 compressor 1136. Working variable D is also applied to the third 4 to 2 compressor 1136. The outputs of the third 4 to 2 compressor 1136 are applied to eighth mux 1154 and ninth mux 1155. Working variables G and B are applied to a tenth mux 1134. The output of the tenth mux 1134 is applied to eighth mux 1154. The output of the

eighth mux 1154 is applied to a second carry look-ahead adder 1162. Chaining variables H7 and H2 are applied to an eleventh mux 1138.

[91] An output of the first carry look-ahead 1160 is applied to a second rotate block 1126. The output of the second rotate block 1126 is applied to the eleventh mux 1138. The output of the eleventh mux 1138 is applied to the ninth mux 1155. The output of the ninth mux 1155 is applied to the second carry look-ahead adder 1162. The output of the second carry look-ahead adder 1162 is applied as shown in Figures 12A and 12B below.

[92] Chaining variables H2, H3, H4, H5 and H8 are applied to a twelfth mux 1140. An output of the twelfth mux 1140 is applied to a thirteenth mux 1156. Working variables A, B, C and H are applied to a fourteenth mux 1144. The output of the fourteenth mux 1144 is applied to a fifteenth mux 1158. Working variables R and S are applied to a fourth 4 to 2 compressor 1142. Functions X[j-7] and X[j-16] are also input to the fourth 4 to 2 compressor 1142. The outputs of the fourth 4 to 2 compressor 1142 are applied to the thirteenth mux 1156 and the fifteenth mux 1158. The outputs of the thirteenth mux 1156 and the fifteenth mux 1158 are applied to a third carry look-ahead adder 1164. The output of the third carry look-ahead adder 1164 is applied as shown in Figures 12A and 12B below.

[93] Figures 11A and 11B are continuations of the flow diagram 1100 of an iteration of a combined MD5, SHA-1 and SHA256 execution, in accordance with one embodiment of the present invention. Referring now to Figure 11A, the output from the first carry look-ahead adder 1160 is also applied to a sixteenth mux 1170. Chaining variable H1 and the initial constants are also applied to the sixteenth mux 1170. The output of the sixteenth mux 1170 is the next H1. The output of the sixteenth mux 1170 is also applied to a seventeenth mux 1182. The output from the first carry look-ahead adder 1160 is also applied to a seventeenth mux 1182. Working variable D is also applied to a seventeenth mux 1182. The output of the seventeenth mux 1182 is the next A.

[94] The output from the first carry look-ahead adder 1160 is also applied to an eighteenth mux 1172. The output from the third carry look-ahead adder 1164 is also applied to the eighteenth mux 1172. Chaining variable H2 and the initial constants are also applied to the eighteenth mux 1172. The output of the eighteenth mux 1172 is the next H2. The output of the eighteenth mux 1172 is also applied to a nineteenth mux 1184. Working variable A and the output of the second carry look-ahead adder 1162 are also applied to the nineteenth mux 1184. The output of the nineteenth mux 1184 is the next B.

[95] The output from the third carry look-ahead adder 1164 is applied to a twentieth mux 1174. Chaining variable H3 and initial constants are also applied to the twentieth mux 1174. The output of the twentieth mux 1174 is the next H3. The output of the twentieth mux 1174 is also applied to a twenty-first mux 1186. Working variable B is also applied to the twenty-first mux 1186. Working variable B is also applied to third rotate block 1180. The output of the third rotate block 1180 is also applied to the to the twenty-first mux 1186. The output of the twenty-first mux 1186 is the next C.

[96] The output from the third carry look-ahead adder 1164 is applied to a twenty-second mux 1176. Chaining variable H4 and the initial constants are also applied to the twenty-second mux 1176. The output of the twenty-second mux 1176 is the next H4. The output of the twenty-second mux 1176 is also applied to a twenty-third mux 1188. Working variable C is also applied to the twenty-third mux 1188. The output of the twenty-third mux 1188 is the next D.

[97] Referring now to Figure 11B, the output from the second carry look-ahead adder 1162 is also applied to a twenty-fourth mux 1202. The output from the third carry look-ahead adder 1164 is also applied to the twenty-fourth mux 1202. Chaining variable H5 and the initial constants are also applied to the twenty-fourth mux 1202. The output of the twenty-fourth mux 1202 is the next H5. The output of the twenty-fourth mux 1202 is also applied to a twenty-fifth mux 1210. The output from the second carry look-ahead adder 1162 is also applied to the twenty-fifth mux 1210. Working variable D is also applied to the twenty-fifth mux 1210. The output of the twenty-fifth mux 1210 is the next E.

[98] The output from the first carry look-ahead adder 1160 is also applied to a twenty-sixth mux 1204. Chaining variable H6 and the initial constants are also applied to the twenty-sixth mux 1204. The output of the twenty-sixth mux 1204 is the next H5. The output of the twenty-sixth mux 1204 is also applied to a twenty-seventh mux 1212. Working variable E is also applied to the twenty-seventh mux 1212. The output of the twenty-seventh mux 1212 is the next F.

[99] The output from the second carry look-ahead adder 1162 is applied to a twenty-eighth mux 1206. Chaining variable H7 and initial constants are also applied to the twenty-eighth mux 1206. The output of the twenty-eighth mux 1206 is the next H7. The output of the twenty-eighth mux 1206 is also applied to a twenty-ninth mux 1214. Working variable F is also applied to the twenty-ninth mux 1214. The output of the twenty-ninth mux 1214 is the next G.

[100] The output from the third carry look-ahead adder 1164 is applied to a thirtieth mux 1208. Chaining variable H8 and the initial constants are also applied to the thirtieth mux 1208. The output of the thirtieth mux 1208 is the next H8. The output of the thirtieth mux 1208 is also applied to a thirty-first mux 1216. Working variable G is also applied to the thirty-first mux 1216. The output of the thirty-first mux 1216 is the next H.

[101] The MD5, SHA-1, SHA256, SHA384 and SHA512 algorithms can also be combined in a single combined execution module. Similar changes needed to combine SHA256, SHA384 and SHA512 algorithms from a SHA256 algorithm-only execution module can be applied to the combined MD5, SHA-1 and SHA256 algorithm execution module to obtain a combined MD5, SHA-1, SHA256, SHA384 and SHA512 algorithm execution module.

[102] As the microprocessor performance becomes ever faster and the microprocessor becomes more densely packed with devices (e.g., transistors, memory, I/O, buffers, etc.), the need for more efficient communication between the microprocessor and peripheral devices such as a crypto co-processor becomes the choking point in overall computer performance. As a result, it has become

advantageous to move more and more of the computer system onto the same IC die as the microprocessor. Crypto-coprocessors have only recently been moved onto the microprocessor chip. An example is the commonly owned U.S. patent application #10/273,718, entitled "Stream Processor with Cryptographic Co-Processor," By Kohn, et al, which was filed on October 18, 2002, which is incorporated by reference herein, in its entirety.

[103] Specifically, U.S. patent application #10/273,718, describes a microprocessor that includes a combination crypto algorithm unit. Figure 12 is a microprocessor die 1300 that includes a multiple crypto algorithm execution module, in accordance with one embodiment of the present invention. The microprocessor die 1300 includes a microprocessor core 1310 and a combination crypto algorithm unit 1320. The combination crypto algorithm unit 1320 can include the capabilities to execute any two or more crypto hash algorithms (e.g., MD5, SHA-1, SHA256, SHA384 and SHA512).

[104] Combining makes more efficient use of the space on a given IC die and specifically combining allows inclusion on the valuable space of a microprocessor die. Combining also allows a more physically compact device thereby reducing processing time (i.e., transit time). An excellent example being that due to the compactness of the design, the combination crypto algorithm unit can be included within the CPU die, therefore reducing data transmission time between the CPU and an external crypto processor. The processing density is increased in that the adders and other devices in the combination crypto algorithm unit are used more often.

[105] With the above embodiments in mind, it should be understood that the invention may employ various computer-implemented operations involving data stored in computer systems. These operations are those requiring physical manipulation of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. Further, the manipulations performed are often referred to in terms, such as producing, identifying, determining, or comparing.

[106] Any of the operations described herein that form part of the invention are useful machine operations. The invention also relates to a device or an apparatus for performing these operations. The apparatus may be specially constructed for the required purposes, or it may be a general-purpose computer selectively activated or configured by a computer program stored in the computer. In particular, various general-purpose machines may be used with computer programs written in accordance with the teachings herein, or it may be more convenient to construct a more specialized apparatus to perform the required operations.

[107] The invention can also be embodied as computer readable code on a computer readable medium. The computer readable medium is any data storage device that can store data that can thereafter be read by a computer system. Examples of the computer readable medium include hard drives, network attached storage (NAS), read-only memory, random-access memory, CD-ROMs, CD-Rs, CD-RWs, magnetic tapes, and other optical and non-optical data storage devices. The computer readable medium can also be distributed over a network coupled computer systems so that the computer readable code is stored and executed in a distributed fashion.

[108] It will be further appreciated that the instructions represented by the operations in any of the figures are not required to be performed in the order illustrated, and that all the processing represented by the operations may not be necessary to practice the invention. Further, the processes described in any of the figures can also be implemented in software stored in any one of or combinations of the RAM, the ROM, or the hard disk drive.

[109] Although the foregoing invention has been described in some detail for purposes of clarity of understanding, it will be apparent that certain changes and modifications may be practiced within the scope of the appended claims. Accordingly, the present embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope and equivalents of the appended claims.

What is claimed is: